Métamodélisation architecturale des procédés logiciels

Fadila Aoussat, Mourad Chabane Oussalah, Mohamed Ahmed-Nacer

Métamodélisation architecturale des procédés logiciels

Fadila Aoussat*, Mourad Oussalah**, Mohamed Ahmed-Nacer***

*Département d'informatique, Université Saad Dahlab Blida,
BP 270, route Soumaa, Blida, Algérie
fadila.aoussat@univ-nantes.fr
**Laboratoire LINA, Université de Nantes
CNRS UMR 6241, 2, Rue de la Houssinière, BP 92208, 44322, Nantes, France
Mourad.oussalah@univ-nantes.fr
***Laboratoire LSI, université des sciences et de la téchnologie Houarie Boumediène,
BP 32, Bab Ezzouar, Algérie.
Anacer@mail.cerist.dz

Résumé. Pour augmenter la réutilisabilité et pour une meilleur modélisation des procédés logiciels, nous proposons d'exploiter le savoir faire et les connaissances acquises durant plusieurs années de recherche pour la modélisation des procédés logiciels. Nous modélisons les procédés logiciels en exploitant les principes des architectures logicielles.

L'objectif de cet article est de présenter un métamodèle générique pour l'extension future du métamodèle SPEM (Software and System Process Engineering Metamodel) avec des concepts architecturaux manquants.

SPEM est un métamodèle adopté par l'OMG pour l'ingénierie des procédés logiciels. Pour la description d'architectures procédés logiciels les concepts architecturaux du métamodèle SPEM sont très insuffisants, en effet, les concepts existants ne permettent pas la description de configurations et de liens explicites pour procédés logiciels ni de les déployer, et par conséquent, de bénéficier des avancées du domaine des architectures logicielles.

1 Introduction

Modéliser de nouveaux procédés logiciels (PLs) en phase avec les nouvelles pratiques, méthodes, outils de développements est une nécessité pour la réussite de projets de développement logiciels. Les Modèles de PLs décrivant l'enchainement d'activités, les responsables, les ressources et les outils utilisés pour la réalisation du produit logiciel doivent refléter et s'adapter à la réalité du développement. Utiliser des modèles de PLs de qualité est alors une garantie pour la réussite des projets de développement logiciels.

Aussi, la modélisation des PLs n'échappe pas aux contraintes de développement auxquels sont soumis les logiciels, modéliser des PLs de qualité dans des délais et à des prix compétitifs reste une priorité. Réutiliser les pratiques et le savoir faire acquis par les précédentes expériences de modélisation et d'exécution de PLs éprouvés est la solution que nous préconisons.

Afin d'augmenter la réutilisabilité de ces connaissances, nous optons pour la modélisation de PLs à base d'architectures logicielles.

Nous proposons une approche de réutilisation de modèles de PLs exploitant le métamodèle SPEM pour la description puis le déploiement d'architectures PLs. Cependant, les concepts architecturaux du métamodèle SPEM [SPEM-OMG (2008)] ne permettent pas la description de PL structuré en architecture logicielle, en effet, l'absence de concepts architecturaux importants, tel que la "configuration", "connecteur explicite" et "style" ne permet pas de manipuler le PL en tant que structure abstraite avant son déploiement final.

Cet article présente la mise en place d'un métamodèle générique basé sur UML regroupant les concepts architecturaux nécessaires pour la description d'architectures PLs, ce métamodèle permettra par la suite l'extension du métamodèle SPEM.

Le reste de l'article est organisé comme suit : la section -2- résume les insuffisances du métamodèle SPEM pour la description d'architectures PLs. La section -3- présente l'idée générale de l'approche proposée pour la modélisation de PLs à base d'architectures logicielles. La démarche adoptée pour l'extension du métamodèle SPEM ainsi que le métamodèle générique pour la description d'architecture PLs sont présentés dans la section-4-. La section-5- conclut notre proposition en résumant les travaux effectués et en énonçant nos perspectives de recherche.

2 Les insuffisances du métamodèle SPEM pour la description architecturale de procédés logiciels

SPEM (System and Software Process Engineering Metamodel) est un métamodèle UML (profil UML) adopté par l'OMG qui décrit les concepts et principes de base pour la modélisation des PLs. Il couvre la description des concepts d'un large éventail de PLs, sans se spécialiser dans un type particulier et en prenant en compte la diversité reconnue des PLs [SPEM-OMG (2008)].

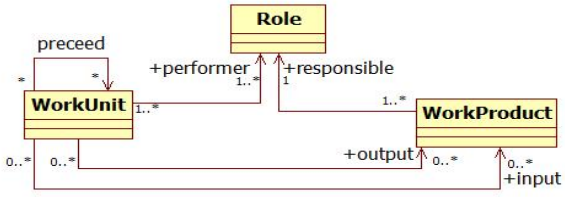

FIG. 1 – *Noyau conceptuel des procédés logiciels.*

Le noyau conceptuel de SPEM est constitué de concepts de base de tout PL (figure-1-). Le PL est un enchainement d'unités de travail. Chaque unité de travail " Work Unit" a besoin de produits en entrée "Work Products" pour fournir des produits en sortie. Une unité de travail est sous la responsabilité d'un rôle "Role".

D'autres concepts procédés tel que "Ressources", "Outils", "Personnel", "Guidance"...etc sont plus au-moins présents selon le type et l'orientation du PL (centré ressources, centré

personnels, centré guidances ...etc). SPEM attribue un stéréotype abstrait "Extensible Element" qui permet de spécialiser différents éléments PLs de manière à permettre la description de différents types de PLs.

SPEM traite la réutilisation des PLs à base de composants à travers le profil "Method Plugin". Ce profil introduit la notion de "Composant procédé" et de réutilisation à base de composants à travers la définition de stéréotypes dédiés à cet effet.

Pour la modélisation de PLs à base de composants, des difficultés d'interconnexion de composants procédés ont été constatées, les problèmes soulevés relève de :

- L'hétérogénéité de la terminologie utilisée pour les "Work Product Ports", car l'assemblage est fait manuellement en faisant une correspondance direct entre les "Work Product Ports" des composants à connecter [SPEM-OMG (2008)].
- La difficulté de gestion des nombres de "Work Product Ports" des composants procédés à connecter, pour les composants de nombre de ports différents[SPEM-OMG (2008)].
- L'assemblage des modèles de PLs à base de composants est fait manuellement, ce qui augmente la dépendance du résultat aux connaissances et à l'expérience du développeur procédé [SPEM-OMG (2008)].

Ces difficultés sont dues à l'absence de certains concepts architecturaux, ainsi, nous remarquons :

- L'absence de "connecteurs" prédéfinis ou explicites : Le connecteur "Work Product Port Connector" est un connecteur implicite, c'est un simple "lien" entre des ports "Work Product Ports". Il est défini pour décrire des liens de précédence ou de délégation sans distinction. Aucun mécanisme de facilitation ou d'adaptation d'assemblage n'est intégré ; son rôle se limite à assurer l'enchainement entre les composants procédés. Aussi, les autres propriétés décrivant le connecteur logiciel (sémantique, évolution, propriétés non fonctionnels ...) en tant que concept architectural ne sont pas prises en compte [Medvidovic et Taylor (1997)].
- L'absence du concept "Connector Role" : Ce qui explique la connexion directe entre "Work Product Ports" à travers des "Work Product Port Conncetors".
- L'absence de contraintes d'assemblage : selon les cardinalités de SPEM, un connecteur peut connecter plusieurs ports sans aucune contrainte.
- L'absence d'abstraction architecturale : La notion de "configuration procédé" est absente dans SPEM. Les contraintes topologiques, ne sont pas prises en compte, ce qui ouvre la porte à des modélisations non cohérentes. Le concept "Configuration Method" est défini comme une "Class", il n'est pas considéré comme un élément du modèle PL, mais un ensemble logique d'éléments "Process Package" et d'éléments "Method Content" [SPEM-OMG (2008)] sans contraintes d'assemblage.
- L'absence de styles architecturaux pour les PLs : La formalisation des structures récurrentes des PLs (tel que les cycles de vie de logiciels) pour la réutilisation à base de composants n'a pas été prise en compte.

"Method Plugin" étant un profil UML, l'absence de consensus pour la modélisation des concepts architecturaux dans UML explique les insuffisances constatées. Aussi peu d'approches pour la modélisation d'architecture PLs ont été proposées par rapport aux autres approches de modélisation, ce qui explique l'absence de certains concepts architecturaux.

3 Notre approche de réutilisation de procédés logiciels à base d'architectures logicielles

Notre approche combine les avancées de deux domaines de recherche qui prônent la réutilisation à large échelle : les ontologies et les architectures logicielles. Notre solution a pour principe l'inférence puis le déploiement d'Architectures PLs [Aoussat et al. (2010)]. L'approche repose sur deux points essentiels :

- L'utilisation d'une ontologie de domaine qui permet de capitaliser le savoir faire et les pratiques éprouvées de ce domaine.
- Réutiliser ces connaissances à travers l'inférence des connaissances qui permettent la description puis le déploiement d'architectures PLs.

Notre but est de proposer des Modèles de PLs de qualité, qui répondent aux demandes spécifiques des développeurs PLs, et cela, en s'inspirant des expériences antérieurs de modélisation et d'exécution de PLs. Notre préoccupation est de résoudre les difficultés de modélisation et d'exécution des PLs. Cette solution est très appropriée pour la modélisation de PLs dynamiques à structures fortement modifiables tel que les PLs incrémentaux, itératifs, hétérogènes ou distribués. La description de modèles de PLs en tant qu'architectures logicielles permet d'augmenter la réutilisabilité des PLs. La modélisation d'Architecture PL permet de manipuler le contenu indépendamment de la structure, et la structure indépendamment de l'implémentation. Cette séparation étant une des caractéristiques principales des architectures logicielles, permettant d'offrir une plus grande flexibilité.

L'article présente une solution aux problèmes rencontrés lors de la conceptualisation de l'ontologie procédé logiciel. L'ontologie proposée repose sur la conceptualisation du métamodèle SPEM, ce choix est justifié par le désire d'exploiter un standard adopté et accepté par la communauté. Aussi, SPEM regroupe les concepts du domaine des PLs indépendamment de leur type, ce qui permet de généraliser l'approche à différents types de procédés. Cependant les concepts architecturaux nécessaires pour la modélisation des architectures PLs sont très insuffisants, tels que les "connecteurs explicites", "configurations" et "styles" pour les PLs qui ne sont pas exploités, d'où la nécessité de les introduire.

4 Metamodèle générique pour l'extension du métamodèle SPEM

4.1 Démarche adoptée pour l'extension du métamodèle SPEM

Les insuffisances en concepts architecturaux étant identifiés, l'étape suivante est l'extension de SPEM. Pour cela, nous adoptons la démarche suivante :

1. Ayant déjà les concepts architecturaux de base des procédés SPEM, nous identifions les autres concepts architecturaux pour la modélisation d'architectures PLs à partir des approches existantes de modélisation de procédés orientés réutilisation [Belkhatir et Estublier (1996)] [Medvidovic et al. (2003)][Choi et Scacchi (2000)][Dami et al. (1998)][Coulette et al. (2000)][Dai et al. (2008)][Borsoi et Becerra (2008)]. Cette étape a comme résultat, l'identification et la formalisation des concepts architecturaux dans le domaine des PLs.

2. Nous explorons les métamodèles d'ADLs (Architectures Description Laguages) existants décris en UML, l'objectif est de s'inspirer des métamodèles formels déjà mis en place pour proposer notre propore métamodèle. L'identification des concepts architecturaux pour PL se base sur l'approche ADL, car les ADLs ont une charge sémantique plus riche que les approches classiques [Belkhatir et Estublier (1996)][Dai et al. (2008)][Borsoi et Becerra (2008)], ils introduisent des concepts architecturaux explicites, des techniques et des outils qui permettent de décrire rigoureusement des architectures logicielles. Cette étape a comme résultat l'introduction des concepts architecturaux manquants, le raffinement de la sémantique des concepts architecturaux existants, puis la formalisation de notre métamodèle d'architecture de PL.

3. Nous explorons les différents types d'approches de modélisation d'ADLs à base d'UML [Medvidovic et al. (2002)]. Dans notre travail, nous étendons un profil existant qui est le profil SPEM. La difficulté de l'extension réside dans le choix et la manière de modéliser ces concepts architecturaux [Sunghwan et al. (2004)][Alti et al. (2010)][Alti et al. (2007)]Le résultat de cette étape est l'extension effective du métamodèle SPEM. Nous identifions les stéréotypes nécessaires pour l'intégration du métamodèle générique de l'étape précédente au métamodèle SPEM. Par soucis d'espace, les résultats de cette étape ne sont pas présentés et feront l'objet d'un autre article.

4.2 Concepts architecturaux des approches existantes de réutilisation de procédés logiciels (PLs)

Pour identifier les concepts architecturaux de PLs des approches existantes Nous explorons les approches existantes de modélisation de PLs à base de composants et d'architectures et nous recoupons leurs interprétations ; ainsi :

- Dans les approches orientées composants, le concept central est le concept "Composant procédé". Le "Composant procédé composite" est défini comme un fragment de PL. Selon le noyau conceptuel des PLs, le composant composite décrit un enchaînement d'activités (enchainement de "WorkUnit").
- Le "Composant procédé élémentaire" décrit une activité élémentaire, l'unité élémentaire du PL étant l'unité de travail "WorkUnit".
- Le concept "Port" (fournit ou requis) est représenté par le "Produit" (Work Product) (en entrée ou en sortie) de l'activité (WorkUnit).
- Le concept "configuration procédé" est défini dans les approches orientées architectures, la "Configuration" représente la structure abstraire du PL.
- Le "Rôle" responsable de l'unité de travail (Work Unit), ainsi que les autres concepts procédés tels que "personnel","ressource","guidance"...etc sont une partie intégrante du Composant Procédé et n'ont pas de correspondances directes avec les concepts architecturaux.

  Le tableau -1 - résume les correspondances effectuées par les approches étudiées :
- Le concept "connecteur" n'a pas eu de vision uniforme. Chaque approche propose sa propre interprétation du connecteur. Cependant l'idée qui se dégage est que le connecteur représente "une dépendance entre activités" soit une dépendance de précédence, soit une dépendance de délégation. Nous remarquons aussi que le connecteur acquière

Concept procédé logiciel.	**Concept architecture logicielle**
Fragment procédé logiciel.	Composant composite [Coulette et al. (2000)] [Dai et al. (2008)].
Activité élémentaire.	Composant élémentaire [Gary et al. (1998)] [Avrilionis et al. (1996)].
Produit(entrée/sortie).	Port (requis/fournis) [SPEM-OMG (2008)] .
Structure procédé logiciel.	Configuration [Choi et Scacchi (2000)][Dai et al. (2008)][Belkhatir et Estublier (1996)].

TAB. 1 – *Correspondance formelle des approches existantes.*

plus d'importance dans les approches les plus récentes et est intégré comme entité de première classe (Tableau-2-).

Approche existantes.	Interprétation du connecteur.	Explicite.
PYNODE [Avrilionis et al. (1996)]	Transfert asynchrone de données.	Non.
APEL [Dami et al. (1998)]	Liens de transfert (control flows ou Data flows).	Non.
RHODES [Coulette et al. (2000)]	Appel de fonction.	Non
App. for software aquisition process architectures [Choi et Scacchi (2000)]	Object qui encapsule des mécanismes d'échange de données et des flux de control	Oui.
Connectors for bridging SP models [Medvidovic et al. (2003)]	Activités d'adaptation de produits spécifiques.	Oui.
SPEM [SPEM-OMG (2008)]	Un "Process Element" qui relie les "Work Product Ports"	Non.
App. based on Evolution process components [Dai et al. (2008)]	Unité de communication	Oui.

TAB. 2 – *Le concept "connecteur" selon les approches existantes.*

4.3 Notre sémantique ajoutée aux concepts architecturaux pour les procédés logiciels

Dans notre approche le connecteur procédé est considéré comme une entité de première classe.

Nous définissons notre connecteur procédé comme une activité qui permet de "faciliter et contrôler" les transitions entre les activités procédés. Contrairement, au "Composant Procédé" le "Connecteur procédé" ne crée pas de nouveaux produits, mais "adapte et contrôle" des produits existants.

La distinction entre Activés de "création" de produit (qui constitueront les composants procédés) et Activités "d'adaptation et de contrôle" de données (qui constitueront les connecteurs procédés) modifie la sémantique des concepts identifiés. Ainsi, notre interprétation des concepts architecturaux des PLs conduit à l'identification de :

Concept procédé logiciel.	**Notre sémantique ajoutée**
Fragment de procédé.	Composant procédé composite : Composé de composants et de connecteurs explicites.
Activité de **"création"** de nouveaux produits logiciels.	Composant procédé élémentaire.
Donnée (en entré /en sortie) d'une activité de création.	Port procédé : peut être un Port flux de donnée ou un Port flux de contrôle.
Structure procédé logiciel.	Configuration procédé : Ensemble de composants procédés et connecteurs procédés respectant des contraintes d'assemblage.
Cycle de vie du logiciel.	Stye procédé : Introduit formellement avec les concepts "Types", invariants et contraintes.
Donnée fournie ou requise (données en entré ou en sortie) d'une activité d'adaptation.	Rôle connecteur : peut être un rôle connecteur "Data Flow" ou un rôle connecteur "Control Flow".
L'activité **"d'adaptation ou de contrôle"** des données.	Connecteurs explicites : Taxonomie de Connecteurs prédéfinies.
Lien de précédence entre Activité de création et Activité d'adaptation.	Attachement : Un lien entre un Port et un Connecteur Rôle de même type.
Lien de délégation entre activités.	Binding : Un lien entre les Ports ou entre Connecteurs Rôles de même type.

TAB. 3 – *Notre correspondance entre concepts de procédés logiciels et concepts d'architectures logicielles.*

- **Composant composite :** Décrit comme un assemblage de composants procédés et de connecteurs explicites.
- **Composant procédé élémentaire :** Décrit un traitement réalisé sur des produits en entrée pour "la création" de nouveaux produits en sortie.
- **Port procédé (Interface du composant) :** L'interface d'un composant est un ensemble de points d'interactions du composant procédé ; elle spécifie les services fournis et requis nécessaires de l'exécution du composant procédé. L'interface du composant procédé est un ensemble de "Ports Procédés", les ports requis correspondent aux "données en entrée" nécessaires à l'exécution du composant procédé, Les ports fournis correspondent aux "données en sortie". Deux types de ports sont définis :
  - Ports flux de données (Data Flow Ports) : Spécifiques aux produits des PLs, Il permettent le transfert des produits logiciels du PL.
  - Ports flux de contrôle (Control flow Ports) : Spécifiques aux flux d'exécution des PLs, Il permettent d'identifier l'ordre et l'état d'exécution du PL.
- **Connecteur Procédé :** Décrit un traitement réalisé sur des produits en entrée afin de les adapter ou les évaluer pour les besoins du composant procédé suivant.
- **Rôles connecteur (Interface connecteur) :** L'interface de connecteur procédé est représenté par de "Rôles Connecteur". De la même manière que les ports procédé, ils représentent les données (le produit ou flux d'exécution) requis ou fournis par les connecteurs

procédé deux types de "Connecteur Rôle" sont définis : les "Data Flow" connecteur Rôle et les "Control flow" connecteur rôle.

- **Binding :** Est un "lien" entre les Ports procédé internes et les Ports procédé externes qui permet de décrire la structure interne de la configuration procédé ou d'un composant composite. De la même manière, le binding des rôles Connecteur permet définir des connecteurs complexes en combinant plusieurs connecteurs procédés. Le binding se fait entre ports procédés de même type.
- **Attachement :** Est une "lien" entre un "Port Procédé" et un "Rôle Connecteur" qui permet de formaliser l'enchainement des composants et des connecteurs procédés. L'attachement se fait entre port et connecteur rôle de même type.
- **Configuration procédé :** Elle décrit l'ensemble logique des composants procédés et des connecteurs procédés en déterminant explicitement les contraintes d'assemblage de la structure procédé. Une configuration peut respecter un style prédéfini tel que le cycle de vie de logiciel ou pas.
- **Style procédé :** Fournit une description partielle de la logique d'assemblage des structures prédéfinies et récurrentes dans les PLs. Le style procédé est introduit formellement, nous définissons les concepts types, invariants et des contraintes topologiques de manière explicite.

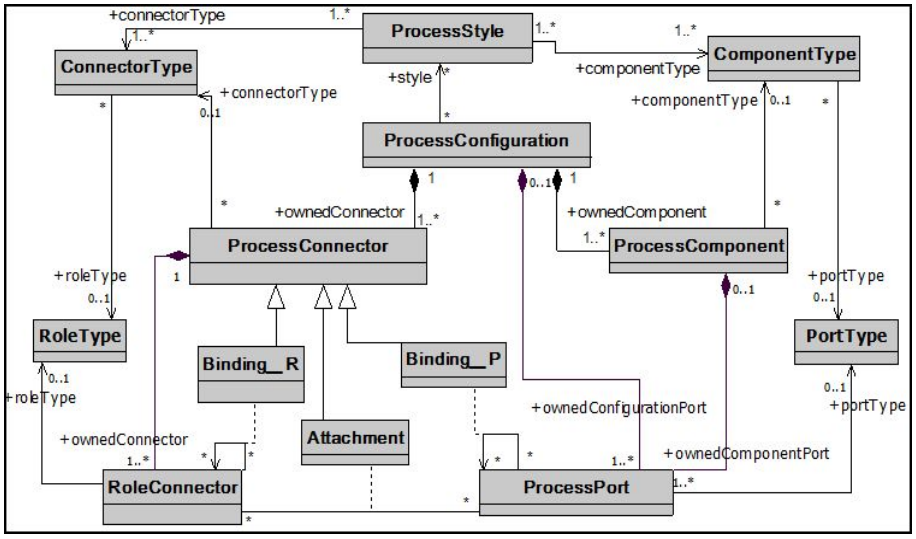

FIG. 2 – *Notre métamodèle générique regroupant les concepts architecturaux des procédés logiciels.*

Nous regroupons Les concepts identifiés en un métamodèle générique basé sur UML. Ce métamodèle est indépendant du métamodèle SPEM, il servira de base de réflexion pour l'extension du métamodèle SPEM. Nous introduisons les concepts "Type Composant", "Type connecteur", "Type Port", "Type Role" pour décrire formellement les styles architecturaux de PLs. Ces concepts sont définis indépendamment des concepts PLs, ils sont introduit pour décrire formellement les styles de PLs.

Ainsi, comme pour les architectures logicielles, une "configuration Procédé" est constituée de "composants procédés" connectés à travers des "connecteurs procédés". La description de la structure interne des "connecteurs procédés" et des "composants procédés" se fait en utilisant des "binding". Le "style procédé" est décrit à travers des associations aux classes "Type" des différents concepts.

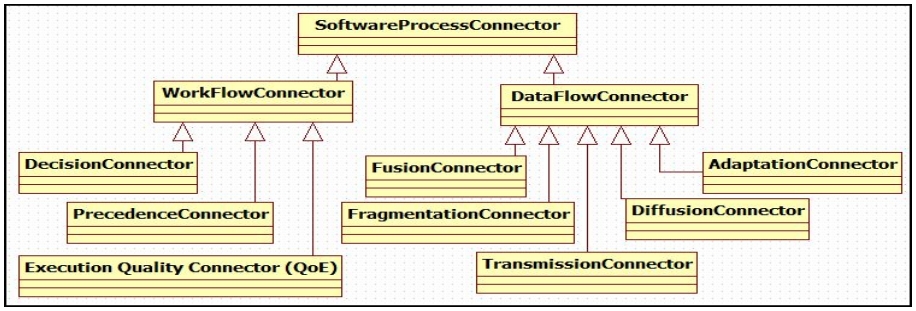

FIG. 3 – *Taxonomie de connecteurs procédés prédéfinis.*

Nous identifions une taxonomie de connecteurs explicites pour la modélisation des architectures PLs. Ces connecteurs procédés nous offrent la possibilité de gérer les interactions indépendamment du type du PL. Ces connecteurs sont très intéressants pour les méthodes agiles respectant des processus où la flexibilité et la dynamicité sont très recherchées.

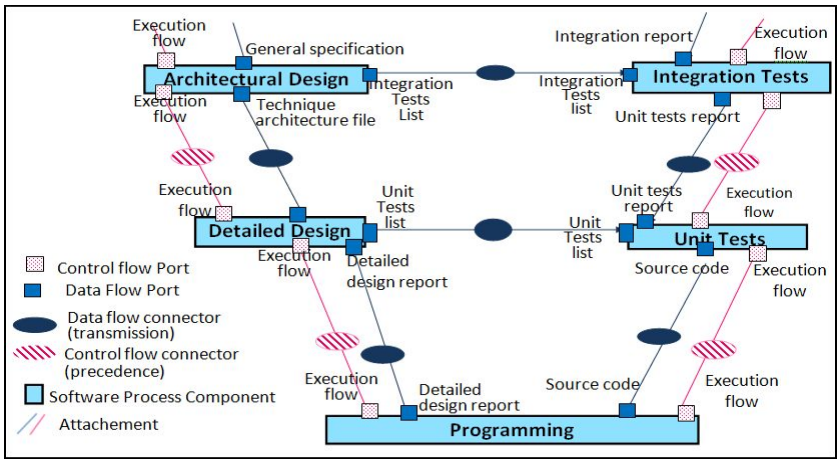

FIG. 4 – *Configuration procédé logiciel respectant le style topologique "cycle de vie en V".*

La figure -4- illustre bien la vue architecturale du PL selon la sémantique adoptée. La configuration procédé, est un assemblage de composants procédés et de connecteurs procédés. Le contrôle de flux est assuré par des connecteurs "Controle Flow", par contre, le transfert des produits est assuré par des connecteurs "DataFlow". Ces deux types de connecteurs ont leur propre type de ports.

5 Conclusion

Cet article présente la mise en place d'un métamodèle générique regroupant les concepts architecturaux de PLs. L'objectif est définir une sémantique pertinente et rigoureuse pour la description puis le déploiement d'architectures PLs. Ce metamodèle est la première étape pour l'extension du métamodèle SPEM en concepts architecturaux manquants. L'extension du métamodèle SPEM est nécessaire pour la mise en place de notre approche de réutilisation de PLs à base d'architectures logicielles, elle concerne le profil "Method plugin" qui est dédié à la réutilisation de PLs.

Afin de proposer ce métamodèle, nous avons, en premier lieu, relevé les insuffisances du métamodèle SPEM, puis identifié les concepts architecturaux PLs des approches existantes. L'analyse de ces approches ainsi que l'étude des métamodèles basés UML proposés pour la description des architectures logicielles nous ont permis de donner une interprétation complète aux concepts Architecturaux PLs, de manière à décrire entièrement une "Architecture PL".

La contribution la plus importante, est la définition de connecteurs explicites spécifiques aux PLs ; ces connecteurs permettent de faciliter, adapter, contrôler les interactions PL. La distinction entre les activités de "création" et les activités "d'adaptation et de contrôle" confère aux PLs une plus grande flexibilité, ce qui est très sollicité dans les nouveaux processus de développement.

D'autres part, la définition de styles architecturaux spécifiques aux PLs contribue significativement à la modélisation et l'exécution des PLs ; ainsi, il est possible, non seulement de réutiliser les bonnes pratiques et les démarches personnelles adoptées par les développeurs procédés (en formalisant ce savoir faire), mais aussi, de combiner, personnaliser, adapter ces pratiques tout en assurant la cohérence du résultat.

Aussi, la séparation des styles "topologique" des styles "d'exécution" nous offre la possibilité de formaliser le style d'exécution de manière explicite et autonome. Cette possibilité est un atout considérable pour assurer la bonne exécution du modèle de PL, et par conséquent, assurer la réussite du projet de développement logiciel.

Références

Alti, A., A. Boukerram, A. Smeda, S. Maillard, et M. Oussalah (2010). Cosabuilder and cosainstantiator : An extensible tool for architectural description. *International Journal of Software Engineering and Knowledge Engineering 20*(3), 423–455.

Alti, A., T. Khammaci, et A. Smeda (2007). Integrating software architecture concepts into the mda platform with uml profile. *Journal of Computer Science 10*, 793–802.

Aoussat, F., M. Ahmed-Nacer, et M. Oussalah (2010). Reusing approach for software processes based on software architectures. In *ICEIS*, pp. 366–369.

Avrilionis, D., N. Belkhatir, et P. Y. Cunin (1996). A unified framework for software process enactment and improvement. In *ICSP '96: Proceedings of the Fourth International Conference on the Software Process (ICSP '96)*, Washington, DC, USA, pp. 102. IEEE Computer Society.

Belkhatir, N. et J. Estublier (1996). Supporting reuse and configuration for large scale software process models. In *ISPW '96: Proceedings of the 10th International Software Process Workshop*, Washington, DC, USA, pp. 35. IEEE Computer Society.

Borsoi, B. T. et J. L. R. Becerra (2008). A method to define an object oriented software process architecture. *Software Engineering Conference, Australian 0*, 650–655.

Choi, J. et W. Scacchi (2000). Modeling and simulating software acquisition process architectures. *Journal of Systems and Software 59*, 343–354.

Coulette, B., T. D. Thu, X. Crégut, et B. T. Dong Thi (2000). Rhodes, a process component centered software engineering environment. In *ICEIS*, pp. 253–260.

Dai, F., T. Li, N. Zhao, Y. Yu, et B. Huang (2008). Evolution process component composition based on process architecture. In *IITAW '08: Proceedings of the 2008 International Symposium on Intelligent Information Technology Application Workshops*, Washington, DC, USA, pp. 1097–1100. IEEE Computer Society.

Dami, S., J. Estublier, et M. Amiour (1998). Apel: A graphical yet executable formalism forprocess modeling. *Automated Software Engg. 5*, 61–96.

Gary, K., T. Lindquist, H. Koehnemann, et J. Derniame (1998). Component-based software process support. *Automated Software Engineering, International Conference on 0*, 196.

Medvidovic, N., P. Grünbacher, A. Egyed, et B. W. Boehm (2003). Bridging models across the software lifecycle. *J. Syst. Softw. 68*, 199–215.

Medvidovic, N., D. S. Rosenblum, D. F. Redmiles, et J. E. Robbins (2002). Modeling software architectures in the unified modeling language. *ACM Transactions on Software Engineering and Methodology 11*(1), 2–57.

Medvidovic, N. et R. N. Taylor (1997). A framework for classifying and comparing architecture description languages. In *ESEC / SIGSOFT FSE*, pp. 60–76.

SPEM-OMG (2008). Software Systems Process Engineering Metamodel, v2.0, Object Management Group OMG.

Sunghwan, R., K. Kyungrae, et J. Taewoong (2004). Architecture modeling language based on uml2.0. In *APSEC '04: Proceedings of the 11th Asia-Pacific Software Engineering Conference*, Washington, DC, USA, pp. 663–669. IEEE Computer Society.

Summary

To increase the reusing and for better software process modeling, Reuse experiences and knowledge capitalized during several decades in the software process modeling field is the proposed solution. We model software processes by exploiting software architecture principles.

The objective of this paper is to present a generic metamodel for the extension of SPEM (System and Software Process Engineering Metamodel) with lacking architectural concepts.

SPEM is a metamodel adopted by the OMG for software process engineering. For the software process architecture description SPEM architectural concepts are very Insufficient, Indeed, the existing concepts disallow describing configurations and explicites links for software processes and their deployment, and consequently, disallow taking advantages from the software architectures field.